HMS: UMA ARQUITETURA PARA AUTOMAÇÃO RESIDENCIAL ABERTA INDEPENDENTE DE TECNOLOGIA DE REDE

Lucas Francisco[1], Kleber Manrique Trevisani[2]

[1]Faculdade de Informática da Universidade do Oeste Paulista (UNOESTE), Presidente Prudente, SP. [2]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP). E-mail: kleber@ifsp.edu.br

RESUMO

House Management System (HMS) é uma arquitetura projetada para automação residencial, que tem por objetivo estruturar a integração de programas de computador para controlar dispositivos eletrônicos através da Internet, a partir de dispositivos móveis. Feitosa Jr. et al. (2010) desenvolveram uma implementação dessa arquitetura utilizando a plataforma JAVA, mas concluíram que o controle de dispositivos eletrônicos de diversos fabricantes e a comunicação com dispositivos utilizando tecnologias de redes de computadores distintas exigiriam um grande esforço de implementação, se forem seguidas as diretrizes da arquitetura. Nesse contexto, este trabalho descreve adequações dessa arquitetura que permitem uma maior abertura no sentido de facilitar a integração com dispositivos eletrônicos e tecnologias de rede distintas. Apresenta também detalhes de implementação que provam a viabilidade dessas adequações.

Palavras-chave: Automação residencial, sistemas distribuídos, arquitetura de software.

HMS: AN OPEN ARCHITECTURE FOR HOME AUTOMATION INDEPENDENT OF NETWORK TECHNOLOGY

ABSTRACT

House Management System (HMS) is an architecture designed for home automation, which aims to structure the integration of computer programs to control electronic devices through the Internet from mobile devices. Feitosa Jr. et al. (2010) developed an implementation of this architecture using the Java platform, but concluded that the control of electronic devices from different manufacturers and the communication with devices using networks computer technologies distinct require a large implementation effort, if followed the guidelines of the architecture. In this context, this paper describes adaptations in this architecture that allow a bigger openness to facilitate the integration between eletronic devices and distinct network technologies. It also presents implementation details to prove the viability of these adjustments.

Keywords: Home automation, distributed systems, software architecture.

INTRODUÇÃO

Há cerca de 20 anos, WEISER (1991) criou o termo computação ubíqua referindo-se a existência de sistemas computacionais nos mais triviais objetos, de forma que o usuário não perceberia a presença da tecnologia nos mais diversificados segmentos. “As tecnologias mais profundas são aquelas que desaparecem: são costuradas no tecido da vida cotidiana até ficarem indistinguíveis dela” (WEISER, 1991).

Porém, para que isso fosse possível, nos dias atuais, muitos avanços foram necessários. A miniaturização dos componentes de hardware, diminuição de custos, a evolução das redes de computadores, dentre outros fatores, permitiram que as idéias de WEISER (1991) fossem passiveis de implementação.

Tais avanços tecnológicos beneficiaram diferentes áreas de atuação, dentre elas a automação residencial. Ela tem como finalidade, proporcionar maior comodidade, conforto e segurança ao usuário. Funcionalidades como manipulação de dispositivos eletrônicos, a partir de qualquer lugar do mundo, através da Internet, são atrativos que tornam a automação residencial um sonho de consumo. No entanto, algumas funções podem ajudar em economia de recursos (água, energia elétrica, gás e etc) e tempo, como por exemplo, realizar atividades que se repetem todos os dias automaticamente (ligar as luzes ao entardecer). No futuro tais vantagens possivelmente tornarão a automação residencial um item importante nas residências e não somente um item de luxo.

A automação residencial não é algo novo. Existem varias tecnologias que atuam neste segmento com importância significativa no mercado, como apresentado na seção 3. No entanto todas possuem arquiteturas fechadas, dificultando a integração de dispositivos desenvolvidos por terceiros. Nesses casos, o usuário fica limitado as funcionalidades disponíveis na tecnologia adquirida, sem poder selecionar um dispositivo de sua preferência ou que apresente menor custo.

Este artigo é composto por sete seções. A seção 1 apresenta as características gerais da arquitetura HMS. A seção 2 trata do objetivo do trabalho. Ela não se encontra próxima a introdução, pois é necessário um conhecimento prévio sobre a arquitetura, antes que os objetivos do trabalho sejam apresentados. A metodologia utilizada durante desenvolvimento do trabalho é tratada na seção 3. Na seção 4 são descritas as principais tecnologias de automação residencial existentes no mercado. A seção 5 apresenta a proposta de mudança estrutural da arquitetura e a seção 6 descreve as alterações de implementação. Finalmente, a seção 7 apresenta as considerações finais e sugestões para trabalhos futuros.

1. FORMULAÇÃO DO PROBLEMA

Visando desenvolver uma arquitetura de automação residencial que permitisse o controle de dispositivos eletrônicos, foi projetada a arquitetura *House Management System* (HMS). Ela vem sendo desenvolvida por alunos do grupo de pesquisa em rede de computadores NetBuilder (NETBUILDER, 2011) em trabalhos de conclusão de curso e projetos de iniciação cientifica.

A arquitetura foi projetada inicialmente por BENTLIN e TREVISANI (2008), ilustrada na Figura 1, para permitir o controle de dispositivos eletrônicos instalados em um local (residência, escritório ou sala de cirurgia, por exemplo) a partir de dispositivos móveis conectados a Internet.

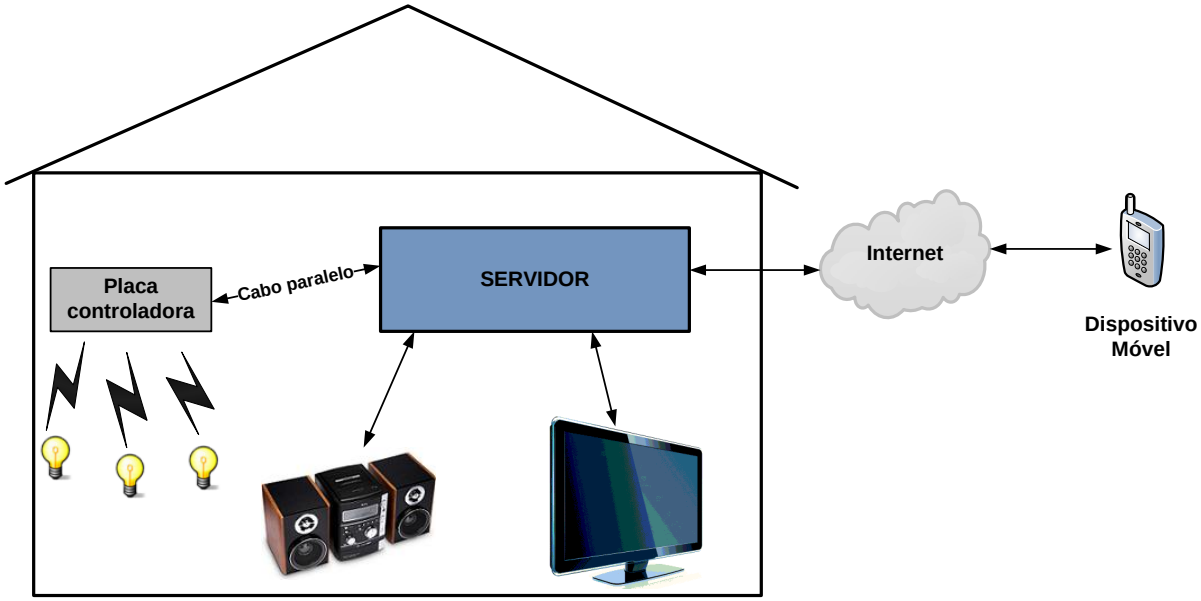

Figura 1. Visão Geral da arquitetura.

O foco do trabalho de BENTLIN e TREVISANI (2008) não era somente o projeto de uma arquitetura de software, mas também a implementação do software. Nesse sentido, foi implementado um servidor, que recebia mensagens via *socket* (TANENBAUM; STEEN, 2007), enviadas por um dispositivo móvel (telefone celular) através de uma aplicação móvel, e comunicava-se com uma placa controladora de lâmpadas através da porta paralela. Os comandos recebidos pelo servidor eram enviados a placa controladora para acender e apagar as lâmpadas. Contudo, concluiu-se que a integração de novos dispositivos sempre demandaria alterações no código fonte do servidor, uma grande desvantagem em se tratando de uma arquitetura de software.

Para solucionar tal problema, Feitosa Jr et al. (2010) adequaram a arquitetura inicialmente desenvolvida por Bentlin e Trevisani (2008) de modo a criar diversos componentes de software com o intuito de minimizar o acoplamento. Desta forma, foi constatada a necessidade de padronizar a comunicação entre os componentes através de protocolos de comunicação. Tais protocolos foram projetados por Feitosa Jr et al. (2010).

A placa controladora, totalmente implementada em hardware, foi substituída por *Device Controllers* (DC), que são programas desenvolvidos para intermediar a comunicação entre o servidor e os dispositivos eletrônicos, de forma a minimizar as alterações no software servidor ao se adicionar um novo dispositivo.

A arquitetura de Feitosa Jr et al. (2010), ilustrada pela Figura 2, é composta por componentes que comunicam-se entre si. Um deles é a aplicação *Mobile House Control* (MHC), que se encontra instalada no dispositivo móvel do usuário. A partir dela é possível controlar dispositivos eletrônicos presentes determinado local (ex: residência). Outro componente é o *House Server* (HS), servidor responsável por receber as requisições de manipulação do MHC e enviá-las aos controladores de dispositivo. Também é função do HS comunicar-se com a Base de Dados (DB) que armazena as informações relevantes utilizadas pelo HS em suas operações. Os *Device Controllers* (DC) são programas embarcados responsáveis pelo controle dos dispositivos (lâmpadas, televisores)

do ambiente. Após a requisição ser enviada do dispositivo móvel (MHC) para o servidor (HS), este a encaminha para o DC que efetivará a manipulação. Há também uma interface com o usuário, o *House Control Center* (HCC), que além de possibilitar a manipulação dos dispositivos, é responsável pelo gerenciamento de diferentes funcionalidades como, por exemplo, gerenciamento de usuários e grupos, definição de permissões de acesso, agendamento de tarefas e definições de perfis de ambiente.

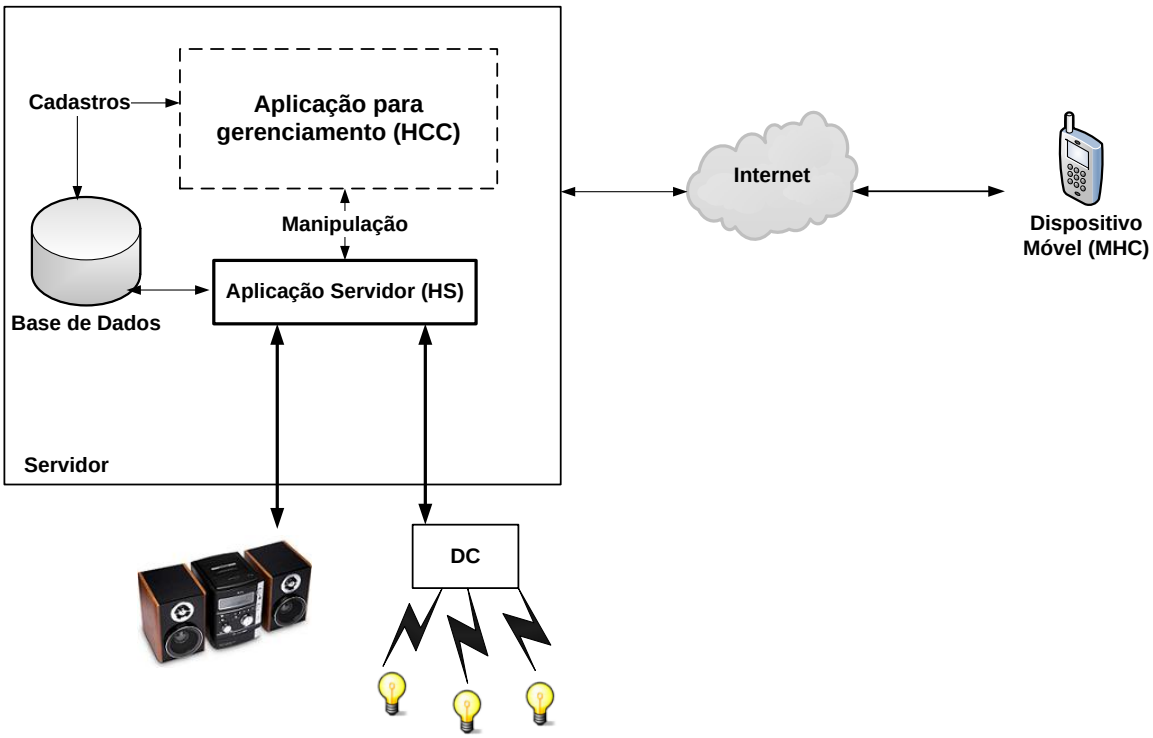

Figura 2. Visão detalhada da arquitetura inicial.

No desenvolvimento do trabalho de Feitosa Jr et al. (2010) constatou-se a necessidade de aumentar o nível de abertura da arquitetura para possibilitar a integração com a maior quantidade de dispositivos eletrônicos da forma mais transparente e aberta possível. Tal conclusão foi alcançada levando em consideração que na arquitetura, forma como foi projetada inicialmente, o hardware dos dispositivos eletrônicos ou seus controladores deviam ser desenvolvidos de acordo com as diretrizes da arquitetura, algo tornava a arquitetura fechada. Adicionalmente, não seria integrar um dispositivo já existente, pois a arquitetura não suportaria o mesmo.

2. OBJETIVOS

A possibilidade de trabalhar com diferentes tecnologias de dispositivos eletrônicos é uma abordagem interessante para uma tecnologia de automação residencial, pois encoraja a adesão de novos fabricantes e facilita a utilização por parte do usuário. Nesse contexto, um dos objetivos deste trabalho é alcançar um nível de abertura (TANEMBAUM; STEEN, 2007) para possibilitar que dispositivos eletrônicos distintos possam ser integrados a estrutura da arquitetura de forma simples e transparente.

Outro objetivo deste trabalho é resolver a dificuldade em realizar a comunicação entre servidor e dispositivos eletrônicos através de diferentes tecnologias de rede (IPv4, ZigBee, Bluetooth, IrDa e etc). Grande parte das tecnologias de automação residencial, inclusive as estudadas neste trabalho e apresentadas na

seção 4, possuem tal limitação. Desta forma, alcançar um alto nível de abertura (TANENBAUM; STEEN, 2007) com relação à integração de redes heterogêneas é uma oportunidade de agregar a arquitetura HMS uma característica que poucas tecnologias possuem. Tal abertura permite que, para que haja a integração com uma nova tecnologia de comunicação, seja necessário apenas uma interface física de comunicação da mesma (placa controladora) instalada no servidor e a implementação de um componente de software no HS que permita a utilização da interface.

Mesmo com tais adequações, é importante destacar que não há intenção de desenvolver um produto comercial. Implementar e testar a viabilidade das especificações da HMS como uma arquitetura que possa ser seguida como base para diferentes implementações de tecnologias de automação residencial é o principal objetivo deste trabalho.

3. METODOLOGIA EMPREGADA

Inicialmente foram realizados estudos detalhados da arquitetura HMS. Foram estudadas sua estrutura, seus componentes e como eles se comunicam através de protocolos, desenvolvidos para padronizar a comunicação entre os componentes da arquitetura de Feitosa Jr et al. (2010). As tecnologias relacionadas também foram foco de estudo para realizar comparativos com a finalidade de agregar qualidade para a arquitetura HMS.

Após a aquisição da fundamentação teórica, foram definidas as alterações necessárias na estrutura da arquitetura para alcançar os objetivos definidos por este trabalho, buscando sempre equilibrar entre viabilidade e boas práticas de codificação.

Para provar a viabilidade das adequações na arquitetura HMS, foram realizadas adequações nos componentes já existentes (desenvolvidos em diferentes trabalhos) e a implementação de novos componentes. Todos os componentes de software foram desenvolvidos utilizando a tecnologia Java (J2SE e J2ME) (SUN MICROSYSTEMS, 2011). Tal escolha se deve ao alto nível de portabilidade da tecnologia Java, o que torna fácil implantar os softwares desenvolvidos em servidores e dispositivos móveis com diferentes tecnologias de hardware e sistemas operacionais totalmente distintos. O servidor que abriga o HS possui sistema operacional Ubuntu. O sistema gerenciador de base de dados escolhido foi o PostgreSQL.

Por fim foram realizados os testes, com as implementações dos componentes de softwares que da arquitetura HMS, para estabelecer um comparativo entre os resultados obtidos e os esperados.

4. TRABALHOS RELACIONADOS

Mesmo sendo uma área de atuação amplamente explorada, apenas nos últimos anos o mercado de automação residencial obteve um crescimento significativo. Tal crescimento se deve pela possibilidade de adquirir produtos de alta tecnologia a um custo relativamente baixo e pela facilidade de inserir tecnologia em dispositivos cada dia menores e mais limitados. O mercado possui diversas tecnologias, entre as mais conhecidas estão o X10, Insteon e o RedEye.

4.1. Tecnologia X10

O X10 é uma das primeiras tecnologias de automação residencial comercializada. Trabalha com mensagens simplificadas comparadas às do HMS e das demais tecnologias, o que torna seu protocolo leve. A comunicação com os dispositivos eletrônicos controlados é realizada, como pode ser visualizado na Figura 3, através da rede elétrica (RYE, 1980). Não é possível utilizar outro meio físico de comunicação, o que torna a tecnologia

limitada, diferente do objetivo deste trabalho para o HMS que é possibilitar a utilização de diferentes tecnologias de comunicação com os dispositivos. Comparado às tecnologias utilizadas atualmente, o X10 se tornou obsoleto, principalmente com relação aos dispositivos eletrônicos que podem ser controlados.

Figura 3. Funcionamento simplificado do protocolo X10.
Fonte: EUROX1O (2011).

4.2. Tecnologia Insteon

O Insteon é uma das mais avançadas tecnologias desenvolvidas para automação residenciais atualmente disponíveis no mercado (INSTEON, 2011).

Suas transmissões de mensagens podem ser realizadas através de RF (rádio frequência) ou rede elétrica, para manter a compatibilidade com X10 (DARBEE, 2005). Com relação à tecnologia de comunicação, o Insteon é limitado, pois não é possível utilizar tecnologias diferentes caso seja necessário, diferente do objetivo deste trabalho para a arquitetura HMS. Os dispositivos, ilustrados na Figura 4, a serem controlados também são desenvolvidos especificamente para o Insteon, o que dificulta a integração com diferentes tecnologias de dispositivos eletrônicos, problema que também fazia parte da arquitetura HMS, porém foi resolvido neste trabalho.

Figura 4. Dispositivos compatíveis com a tecnologia Insteon.
Fonte: Inteon (2011).

Toda a dificuldade de integração (tecnologias de rede e de dispositivos) ocorre devido à estrutura do Insteon ser toda baseada em licenças proprietárias (INSTEON, 2011) incluindo seus softwares e dispositivos eletrônicos. Tal abordagem dificulta o desenvolvimento de melhorias ou adaptações por terceiros, o que reduz o nível de abertura (TANENBAUM; STEEN, 2007).

4.3 Tecnologia RedEye

O RedEye é uma tecnologia que surgiu recentemente no mercado e vem ganhando espaço dentre as demais tecnologias de automação residencial. Ele possui três tipos de hardware (*RedEye Mini, Dock e Pro*) que são utilizados juntamente com um software desenvolvido para dispositivos móveis que

possuam iOS (*iPhone, iPod, iPad*) (THINKFLOOD, 2011).

O RedEye trabalha com duas tecnologias de comunicação, infravermelho e WiFi. O infravermelho é utilizado na comunicação com os dispositivos a serem controlados (televisor, rádio). No caso *RedEye Mini* o hardware é acoplado diretamente ao dispositivo móvel transformando o mesmo em um controle remoto. No *RedEye Dock* e *Pro*, o hardware desenvolvido trabalha como um controlador de dispositivos. O dispositivo móvel que contém o software se comunica com o hardware controlador através de WiFi e o hardware manipula o dispositivo eletrônico através de infravermelho. Na Figura 5 pode ser visualizado o funcionamento simplificado da tecnologia RedEye utilizando o hardware *RedEye Dock*.

Figura 5. Funcionamento simplificado da tecnologia RedEye.

Durante o estudo da tecnologia foi possível constatar, que mesmo com a facilidade de integração de diferentes dispositivos, qualquer um que receba requisições através de infravermelho, há uma dificuldade evidente de integração com diferentes tecnologias de comunicação, sendo possível somente utilizar IPv4 e infravermelho (THINKFLOOD, 2011).

4.4 Comparação da arquitetura HMS com trabalhos relacionados

Ao compararmos a arquitetura HMS às tecnologias relacionadas nas seções 4.1, 4.2 e 4.3 podemos obter alguns comparativos que ajudaram no desenvolvimento da arquitetura a fim de torná-la aberta com relação a tecnologias de rede e de dispositivos eletrônicos.

A X10 por se tratar de uma tecnologia obsoleta se mostra inflexível tanto para a utilização de diferentes tecnologias de comunicação, somente corrente elétrica (RYE, 1980) no caso, quanto na integração com diferentes dispositivos eletrônicos, que devem ser desenvolvidos de forma específica para a arquitetura. Neste trabalho, os principais objetivos é tornar a HMS uma arquitetura aberta de forma que a integração com diferentes tecnologias de dispositivos eletrônicos e de comunicação seja algo simplificado, desde que as especificações da arquitetura sejam seguidas.

A tecnologia Isteon, apesar de difundida quando o assunto de automação residencial, ela é totalmente fechada. Tal abordagem desencoraja o desenvolvimento de novos dispositivos compatíveis e impossibilita a integração com dispositivos desenvolvidos anteriormente, pois todos os dispositivos compatíveis são desenvolvidos pela empresa que desenvolveu a tecnologia, de forma fechada (INSTEON, 2011), enquanto na HMS a intenção é justamente atrair o desenvolvimento de diferentes dispositivos que possam ser integrados facilmente ao sistema. Outro problema é que por utilizar uma tecnologia de comunicação própria (INSTEON, 2011) (que inclusive tem sua própria estrutura de pacotes), se torna impossível optar

por qual tecnologia de comunicação utilizar para conectar o sistema aos dispositivos. Neste trabalho, o objetivo é definir a arquitetura HMS de forma que seja possível escolher, tanto do desenvolvedor do dispositivo, quanto para o usuário de qualquer sistema baseado há arquitetura, de qual tecnologia de comunicação utilizar nas integrações.

A tecnologia RedEye é a mais recente, e a que possui a integração com dispositivos mais facilitada. Basta que o dispositivo se comunique através de infravermelho. Porém, a integração com tecnologias de comunicação é tão limitada quanto a das demais tecnologias estudadas (THINKFLOOD, 2011), sendo possível utilizar somente Wifi e infravermelho, sem que haja a possibilidade de integrar novas tecnologias, enquanto o objetivo da arquitetura HMS é possibilitar a integração com a maior quantidade de tecnologias de comunicação possíveis.

5. ALTERAÇÕES NA ARQUITETURA

A proposta deste trabalho é alcançar um nível de abertura satisfatório com relação à integração de diferentes tecnologias de comunicação e de dispositivos eletrônicos. Para que isso seja possível foram necessárias alterações na estrutura da arquitetura HMS, como pode ser visualizado na Figura 6.

Um fator limitante para a arquitetura projetada por Feitosa Jr et al. (2010) é a dificuldade de trabalhar com tecnologias de rede heterogêneas. A integração com novas tecnologias de rede era possível, porém o esforço de implementação inviabilizaria a utilização de outras tecnologia de comunicação, diferentes de IPv4.

Visando solucionar este problema foram realizadas modificações na arquitetura para desacoplar do HS a parte do código responsável por enviar e receber mensagens de rede. Na proposta deste trabalho, essa responsabilidade é delegada a componentes de software específicos, chamados controladores de rede. É necessário um controlador para cada tecnologia utilizada.

Para garantir uma padronização no desenvolvimento dos componentes controladores de rede, foi implementada uma classe abstrata (HORSTMANN, 2004) que define as características e comportamentos necessários aos controladores de rede. Há também uma implementação no HS seleciona o controlador de rede para qual a mensagem será enviada. Essa seleção depende da identificação da rede em que o dispositivo eletrônico se encontra. A abordagem adotada facilita a integração de novas tecnologias de rede, pois basta desenvolver e integrar um novo componente controlador com implementações específicas da tecnologia desejada. Desta forma é possível realizar comunicação entre o HS e os DCs através de redes totalmente heterogêneas, como por exemplo, IPv4, ZigBee, Bluetooth dentre outras.

Além do controlador de rede, o servidor deve possuir uma interface física (placa controladora) (TANENBAUM, 2003; KUROSE; ROSS, 2006).

Outro problema a ser solucionado é a forma como a arquitetura desenvolvida por Feitosa Jr et al. (2010) lida com a integração de novos dispositivos. Todo novo controlador de dispositivo deve ser desenvolvido baseado nos padrões da arquitetura, para garantir a compatibilidade entre servidor e DCs. Normalmente os DCs suportam a execução de programas de computador embarcados, porém isso não ocorre para a maioria dos dispositivos eletrônicos disponíveis atualmente. Outro ponto negativo da abordagem de FEITOSA JR et. al. (2010) é que os fabricantes dificilmente realizariam alterações em seus produtos para se adequar a arquitetura HMS. Tais problemas inviabilizam tanto o desenvolvimento de novos dispositivos eletrônicos compatíveis, quanto a

integração de dispositivos já existentes. Desta forma, dificilmente fabricantes se sentiriam encorajados a realizar implementações compatíveis com a arquitetura HMS.

A solução escolhida para tal dificuldade foi utilizar tradutores de tecnologia. Os tradutores possuem a função de padronizar a comunicação entre o HS e os DCs de forma leve e independente. Eles devem agregar funções específicas de tradução e não devem fazer mais do que isso. No caso deste trabalho, foi adotada a abordagem de utilização de *plugins,* ilustrado na Figura 6. Um exemplo de tradução seria padronizar mensagens enviadas por um dispositivo de um fabricante qualquer, de forma que o HS reconheça o formato de mensagem.

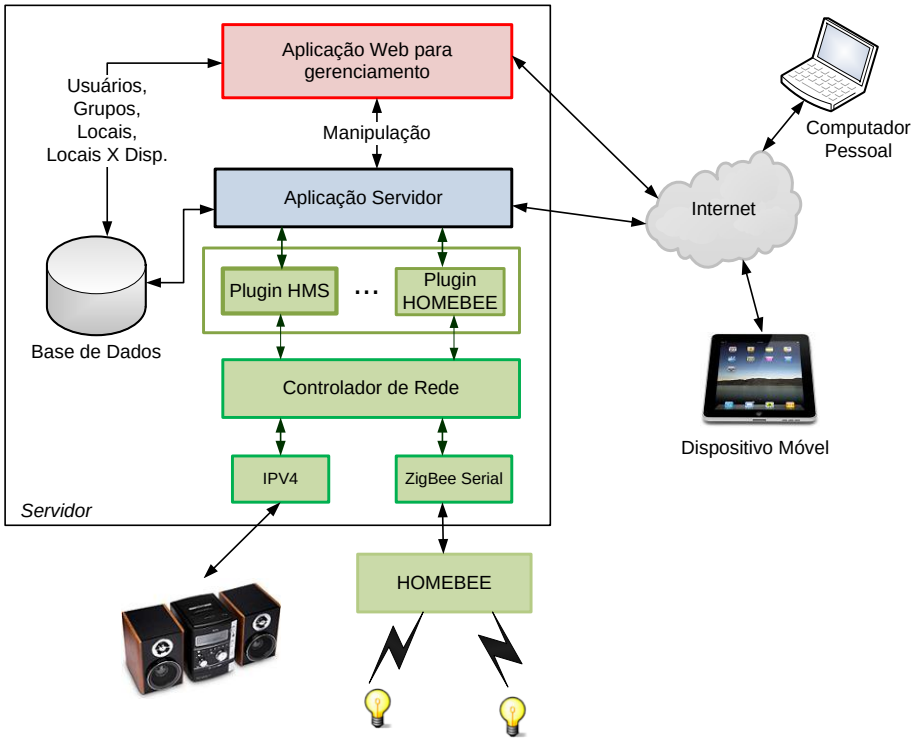

Figura 6. Visão geral da arquitetura após alterações.

6. DETALHES DE IMPLEMENTAÇÃO

Feitosa JR et al. (2010) também desenvolveram um software para provar a viabilidade da arquitetura proposta por eles. Além disso definiram protocolos para padronizar a comunicação entre os componentes de software da arquitetura.

Para provar a viabilidade das alterações propostas por este trabalho, foram necessárias adequações no software já existente. Também foi necessário desenvolver novos componentes de software. Alterações na base de dados também foram necessárias para possibilitar a utilização de agendamentos de tarefas e definição de perfis preferência do usuário.

6.1. Controladores de rede

Tecnicamente, cada controlador de rede é uma classe Java implementada na aplicação HS. Para padronizar o desenvolvimento dos controladores, foi implementada uma classe abstrata chamada *NetWorkController*. Como ilustrado na Figura 7, todos os controladores de rede devem herdar (HORSTMANN, 2004) as propriedades e características da classe *NetWorkController,* o que garante a padronização no desenvolvimento dos controladores de rede.

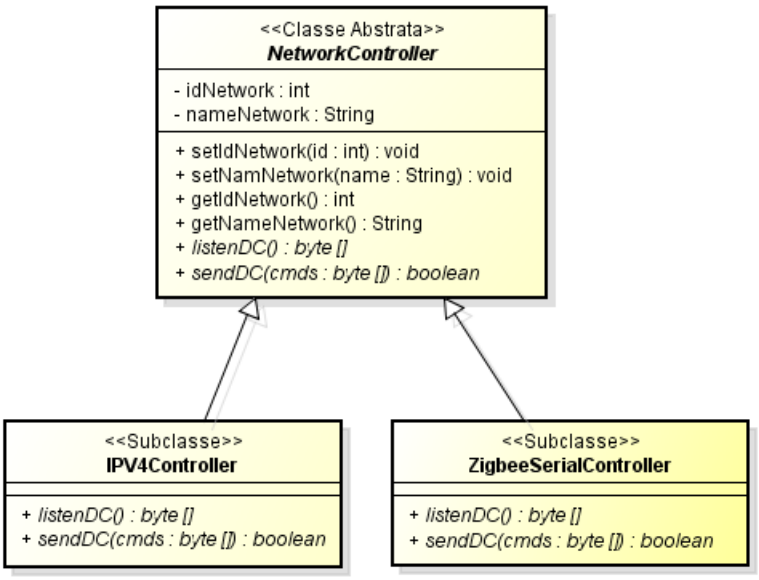

Figura 7. Diagrama UML de classes do padrão de controladoras de rede.

Os controladores de rede diferenciam-se dos demais apenas por detalhes de codificação específicos de cada tecnologia de rede, como por exemplo, endereçamento e tamanho do quadro.

6.2 Implementação dos *plugins*

Um *plugin*, também conhecido como *add-in* ou *add-on*, é definido como um pequeno software que pode ser facilmente acoplado programas de computadores com a finalidade de agregar funcionalidades específicas (PC MAGAZINE, 2011; MERRIAM-WEBSTER, 2011). Tal abordagem geralmente é adotada para permitir que desenvolvedores externos possam agregar funcionalidades específicas a outro software, sem a necessidade de alterá-lo.

Os *plugins,* na estrutura da arquitetura HMS, são componentes de software desenvolvidos para prover a tradução na troca de mensagens entre o HS e os DCs. Tal abordagem permite que mesmo tecnologias em que não exista mais perspectiva de novas atualizações possam ser integradas a estrutura da arquitetura HMS, bastando apenas que o fabricante desenvolva um *plugin* que permita a compatibilidade entre seu dispositivo e um software desenvolvido de acordo com a arquitetura HMS. A Figura 8 ilustra o funcionamento dos *plugins* e a forma como as mensagens são traduzidas por ele. A mensagem enviada por um DC chega de forma incompreensível para o HS. O HS então identifica o *plugin* responsável e solicita a tradução da mensagem recebida para o formato compreendido pelo HS. Caso haja a necessidade de devolver uma resposta ao DC, o HS envia a mensagem ao *plugin* vinculado ao DC, a mensagem é traduzida de forma que o DC compreenda e logo em seguida a mensagem é enviada para o DC.

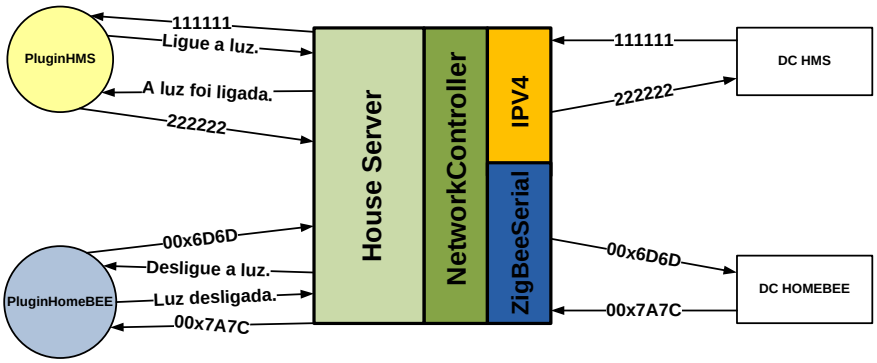

Figura 8. Exemplo de funcionamento dos *plugins*.

O vínculo entre DCs e *plugins* é realizado através de cadastro na base de dados. Tal abordagem facilita a integração de novos *plugins* e posteriormente sua utilização nas traduções de mensagens.

Uma das funcionalidades mais importantes implementada nos *plugins* é a capacidade que os *plugins* têm de informar seus dados para se cadastrar na base de dados, permitindo uma integração mais simplificada. Ao ser iniciado pelo HS, o *plugin* informa seus dados, e caso o mesmo não possua cadastro na base de dados, ele é cadastrado. A Figura 9 ilustra o trecho de código de um plugin que anuncia sua presença no sistema.

```
19  public class Main {
20
21      static final int PORT = 2000;
22      public static void main(String[] args) {
23      ServerPlugin sp = new ServerPlugin(PORT);
24      pacotePlugin p;
25      /*pacote com informações do plugins*/
26      p = new pacotePlugin(PORT, "Plugin HMS", "", "1.0",
27                           "NetBuilder","PluginHMS.jar");
28      /*avisa quem é para o HS - modo "on the fly"*/
29      try
30      {
31          Socket s = new Socket("localhost",1250);
32          DataInputStream input;
33          input = new DataInputStream(s.getInputStream());
34          DataOutputStream output;
35          output = new DataOutputStream(s.getOutputStream());
36          output.writeUTF(p.getP());
37          String response = input.readUTF();
38          output.writeUTF(p.getP());
39          System.out.println(response);
40          response = input.readUTF();
41          System.out.println(response);
42          sp.start();
43      }
44      catch (IOException ex)
45      {
46          System.out.println("ERRO: "+ex.getMessage());
47      }
```

Figura 9. Código fonte da solicitação de cadastro do *plugin.*

Foi definido que todos os *plugins* (arquivos executáveis desenvolvidos utilizando a linguagem Java), utilizados pelo HS, devem estar em um diretório específico do servidor. Quando a aplicação HS é iniciada, ela executa todos os *plugins* presentes na pasta pré-definida, como ilustrado na linha 311 e 312 da Figura 10.

```
305     private ArrayList <Process> process;
306     //para utilizar os plugins de modo "on the fly"
307     public void loadPlugins()
308     {
309         ProcessBuilder builder;
310         Process p;
311         String spath = "/home/netbuilder/Apps/plugins";
312         File path = new File(spath);
313         if(path.isDirectory())
314         {
315             for(File obj : path.listFiles())
316             {
317                 builder = new ProcessBuilder("java", "-jar",obj.getName());
318                 builder.directory(path);
319                 try
320                 {
321                     p = builder.start();
322                     process.add(p);
323                 }
324                 catch (IOException ex)
325                 {
326                     System.out.println("ERRO: "+ex.getMessage());
327                 }
328             }
329         }
330     }
```

Figura 10. Código fonte para iniciar todos os *plugins.*

Para que a abertura com relação a integração de dispositivos fosse completa, o desenvolvimento dos *plugins* por terceiros deveria ser simples. No software desenvolvido para a prova de viabilidade, há um conjunto de classes que devem ser utilizadas por qualquer um que queira construir de um *plugin* evitando reescrita de código

Para desenvolver um *plugin* é necessário que o desenvolvedor conheça previamente o formato de mensagens trocadas entre o HS e os DCs, pois existem classes Java que definem os padrões de mensagens (*pacote.java, pacoteAU.java, pacoteDB.java, pacoteDC.java* e *pacotePlugin.java).* Portanto uma implementação da arquitetura HMS deve publicar explicitamente os formatos de mensagens definidos, para que outros desenvolvedores possam desenvolver *plugins* compatíveis com menos esforço de implementação.

Existem ainda classes que definem comportamentos que todos os *plugins* devem possuir (*Monitor.java, ServerPlugin.java* e *message.java*), como, por exemplo, enviar dados para verificar a necessidade de se cadastrar na base de dados ou monitorar a situação do HS - caso ele não esteja ativo, o *plugin* pode *se* desligar automaticamente. Tais classes devem ser adicionadas no projeto de qualquer novo *plugin* desenvolvido para garantir que seu funcionamento seja satisfatório.

6.3. Testes com dispositivos

Para provar a viabilidade das alterações da arquitetura foi utilizado um DC, desenvolvido por Feitosa Jr. et al. (2010) que se comunica através da tecnologia IPv4. Também foi utilizada uma placa Homebee, que funciona como um DC e controla duas lâmpadas. Ela se comunica através de rede ZigBee (ZIGBEE, 2011).

Essas duas tecnologias foram escolhidas propositalmente, por possuírem diferenças no formato de mensagem e endereçamento na tecnologia de transmissão utilizada, por ser uma situação extrema a ser testada quanto a adequação da arquitetura.

Para cada uma das tecnologias foi desenvolvido um *plugin* e um controlador de rede. Neste trabalho foram desenvolvidos um *plugin* para o DC desenvolvido por Feitosa Jr. et al. (2010) e um controlador de rede da tecnologia IPv4. Para a placa Homebee, foram desenvolvidos (em outro trabalho) um *plugin,* capaz de realizar a tradução do seu formato de mensagens para o formato do software desenvolvido, e um controlador de rede para a

tecnologia de comunicação ZigBee. Alterações na modelagem da base de dados tornaram possível a implementação, em software, do agendamento de tarefas e definição de perfis de usuário, algo desejável para uma arquitetura de automação residencial.

7. CONSIDERAÇÕES FINAIS

Tratando-se de uma arquitetura para automação residencial, a facilidade para integrar novas tecnologias de dispositivos eletrônicos e de trabalhar com tecnologias de comunicação heterogêneas pode ser algo interessante. A arquitetura HMS alcançou um nível de abertura que a diferencia da grande maioria das tecnologias presentes no mercado, e ao termino do trabalho, foi possível constatar que o nível de abertura almejado pelo trabalho foi alcançado de forma satisfatória.

Durante definição das adequações necessárias, um dos principais desafios foi manter os componentes que não seriam alterados funcionando. Um exemplo que pode-se destacar é que a aplicação MHC (aplicação móvel) não sofreu nenhuma alteração e continuou funcionando da maneira esperada, mesmo após todas alterações na implementação.

Outro problema encontrado foi a dificuldade de manter a consistência da base de dados em situações de adversidades, como por exemplo, uma quedas de energia. Muitas vezes a situação do ambiente controlado não é equivalente as informações gravadas na base de dados. Tal problema ainda não foi solucionado. No entanto, um controle de tolerância a falhas e de manutenção da consistência entre a base de dados e a situação real do ambiente poderia solucionar tal problema. Uma forma de implementar tal controle seria através da análise do log do sistema.

Não é possível afirmar que todas as tecnologias de dispositivos e de comunicação possam ser integradas a um software baseado nessa arquitetura, porém com as implementações realizadas foi possível constatar que as integração de novos dispositivo eletrônicos e rede de comunicação tornam-se mais simples, pois o nível de abertura proposto pelo trabalho foi alcançado de forma satisfatória.

Alguns trabalhos futuros podem agregar funcionalidades interessantes para a implementação da arquitetura HMS, aumentando sua viabilidade.

Dentre as possíveis melhorias, uma delas seria a utilização dos registros de log para uma analise de perfil de usuário, haja vista que todas as atividades realizadas pelo HS são registradas na base de dados em forma de log, porém não são aproveitadas. Tal analise possibilitaria criar perfis de preferência de usuários, onde a implementação da arquitetura se adequaria aos padrões de manipulação do usuário. Desta forma, quando o usuário, por exemplo, ligasse seu televisor durante toda a semana após chegar em casa do trabalho, o HS questionaria a possibilidade de tornar ou não aquela atividade um permanente. Caso o usuário escolhesse que sim, todos os dias, no mesmo horário a ação de ligar o televisor seria repetida. Tal abordagem pode ser implementada através de técnicas de estatística e inteligência artificial.

Além das melhorias citadas acima, testes de integração com novos controladores de dispositivos, como Arduino e RedEye, e com novas tecnologias de rede, como Bluetooth (BLUETOOTH, 2011) e IrDa (IRDA, 2011), possibilitariam verificar a capacidade da arquitetura de adequar-se a novas tecnologias.

REFERÊNCIAS

BENTLIN, E. ; TREVISANI, K. **MHC: Um sistema para controle de locais utilizando dispositivos móveis.** 2008**.** Trabalho de conclusão de curso (Bacharelado em Ciência da Computação) – Faculdade de Informática de Presidente Prudente – UNOESTE, Presidente Prudente.

BLUETOOTH. **Bluetooth Technical Information. 2011.** Disponível em: <http://www.bluetooth.com/Pages/Tech-Info.aspx>. Acesso em: 15 dez. 2011

DARBEE, P. **Insteon: The details**. Manual de referência – Smarthome technology, 2005.

EUROX10. **O que é o X10?** 2011. Disponível em: <http://www.eurox10.com/Content/X10Information.htm>. Acesso em: 12 nov. 2011.

FEITOSA Jr. et al. **HMS: Um sistema aberto para automação residencial.** Presidente Prudente: Colloquium Exactarum vol. 2 No. 2 – Unoeste, 2010.

HORSTMANN, C. **Big java**. Porto Alegre: Bookman, 2004.

INSTEON. **"Technical Information"**. 2011. Disponível em: <http://www.insteon.net/about-technicalinfo.html>. Acesso em: 15 nov. 2011.

IRDA. **About IrDa.** Disponível em: < http://www.irda.org/displaycommon.cfm?an=1 >. Acesso em: 15 dez. 2011

KUROSE, J.; ROSS, K. **Redes de Computadores e a Internet: Uma abordagem Top-down**. 3ª ed. São Paulo: Pearson Addison Wesley, 2006.

MERRIAM-WEBSTER. **Definition of PLUGIN**. Disponível em: <http://www.merriam-webster.com/dictionary/plugin>. Acesso em: 15 dez. 2011.

NETBUILDER. **Diretório de grupos de pesquisa do Brasil: Grupo NetBuilder**. 2011. Disponível em: < http://dgp.cnpq.br/buscaoperacional/detalhegrupo.jsp?grupo=5067103GALF20A>. Acesso em: 16 dez. 2011.

PC MAGAZINE. **Definition of: plug-in**. 2011. Disponível em: <http://www.pcmag.com/encyclopedia_term/0,2542,t=plug-in&i=49395,00.asp> . Acesso em: 20 mar. 2011.

RYE, D. **Technical notes – The X10 Powerhouse**. Manual de referência – Powerhouse, 1980.

SUN MICROSYSTEMS. **Java™ Platform, Standard Edition 6 API Specification**. 2011. Disponível em: < http://java.sun.com/reference/api/>. Acesso em: 20 nov. 2011.

TANENBAUM, A; STEEN, M. **Distributed systems**. New York: Prentice Hall, 2007.

TANENBAUM, A. **Redes de computadores**. 4ª ed. Rio de Janeiro: Campus, 2003.

THINKFLOOD. **RedEye user manual. 2011.** Disponível em: <http://thinkflood.com/support/redeye/redeye-user-manual>. Acesso em: 5 nov. 2011

ZIGBEE. **ZigBee specifications.** 2011. Disponível em: < http://www.zigbee.org/Specifications.aspx >. Acesso em: 10 nov. 2011.

WEISER, M. **"The computer for the 21st century".** Scientific American, v. 265, n. 3, p. 66-75, 1991. http://dx.doi.org/10.1038/scientificamerican0991-94

Recebido em: 05/06/2013

Revisado em: 25/06/2013

Aceito em: 01/07/2013